8.1 Introduction

Images are signals with two spatial dimensions. This chapter deals with signals of arbitrary dimensions. This generalization is very useful because computer vision is not restricted solely to 2-D signals. On the one hand, higher-dimensional signals are encountered. Dynamic scenes require the analysis of image sequences; the exploration of 3-D space requires the acquisition of volumetric images. Scientific exploration of complex phenomena is significantly enhanced if images not only of a single parameter but of many parameters are acquired. On the other hand, signals of lower dimensionality are also of importance when a computer vision system is integrated into a larger system and image data are fused with time series from point-measuring sensors.

8.2 Continuous signals

8.2.1 Types of signals

An important characteristic of a signal is its *dimension*. A zero-dimensional signal results from the measurement of a single quantity at a single point in space and time. Such a single value can also be averaged over a certain time period and area. There are several ways to extend a zero-dimensional signal into a 1-D signal (Table 8.1). A *time series* records the temporal course of a signal in time, while a *profile* does the same in a spatial direction or along a certain path.

A 1-D signal is also obtained if certain experimental parameters of the measurement are continuously changed and the measured parameter is recorded as a function of some control parameters. With respect to optics, the most obvious parameter is the wavelength of the electromagnetic radiation received by a radiation detector. When radiation is recorded as a function of the *wavelength*, a *spectrum* is obtained. The

Table 8.1: *Some types of signals g depending on D parameters*

D	Type of signal	Function
0	Measurement at a single point in space and time	g
1	Time series	$g(t)$
1	Profile	$g(x)$
1	Spectrum	$g(\lambda)$
2	Image	$g(x,y)$
2	Time series of profiles	$g(x,t)$
2	Time series of spectra	$g(\lambda,t)$
3	Volumetric image	$g(x,y,z)$
3	Image sequence	$g(x,y,t)$
3	Hyperspectral image	$g(x,y,\lambda)$
4	Volumetric image sequence	$g(x,y,z,t)$
4	Hyperspectral image sequence	$g(x,y,\lambda,t)$
5	Hyperspectral volumetric image sequence	$g(x,y,z,\lambda,t)$

wavelength is only one of the many parameters that could be considered. Others could be temperature, pressure, humidity, concentration of a chemical species, and any other properties that may influence the measured quantity.

With this general approach to multidimensional signal processing, it is obvious that an image is only one of the many possibilities of a 2-D signal. Other 2-D signals are, for example, time series of profiles or spectra. With increasing dimension, more types of signals are possible as summarized in Table 8.1. A 5-D signal is constituted by a *hyperspectral* volumetric image sequence.

8.2.2 Unified description

Mathematically, all these different types of multidimensional signals can be described in a unified way as continuous scalar functions of multiple parameters or generalized coordinates q_d as

$$g(\boldsymbol{q}) = g(q_1, q_2, \ldots, q_D) \quad \text{with} \quad \boldsymbol{q} = [q_1, q_2, \ldots, q_D]^T \tag{8.1}$$

that can be summarized in a D-dimensional *parameter vector* or generalized coordinate vector $\boldsymbol{q}$. An element of the vector can be a spatial direction, the time, or any other parameter.

As the signal g represents physical quantities, we can generally assume some properties that make the mathematical handling of the signals much easier.

Continuity. Real signals do not show any abrupt changes or discontinuities. Mathematically, this means that signals can generally be regarded as arbitrarily often differentiable.

Finite range. The physical nature of both the signal and the imaging sensor ensures that a signal is limited to a finite range. Some signals are restricted to positive values.

Finite energy. Normally a signal corresponds to the amplitude or the energy of a physical process. As the energy of any physical system is limited, any signal must be square integrable:

$$\int_{-\infty}^{\infty} |g(\boldsymbol{q})|^2 \, \mathrm{d}^D q < \infty \tag{8.2}$$

With these general properties of physical signals, it is obvious that the continuous representation provides a powerful mathematical approach. The properties imply, for example, that the Fourier transform (Section 8.6) of the signals always exist.

Depending on the underlying physical process the observed signal can be regarded as a stochastic signal. More often, however, a signal is a mixture of a deterministic and a stochastic signal. In the simplest case, the measured signal of a deterministic process g_d is corrupted by additive *zero-mean homogeneous noise*. This leads to the simple signal model

$$g(\boldsymbol{q}) = g_d(\boldsymbol{q}) + n \tag{8.3}$$

where n has the *variance* $\sigma_n^2 = \langle n^2 \rangle$. In most practical situations, the noise is not homogeneous but rather depends on the level of the signal. Thus in a more general way

$$g(\boldsymbol{q}) = g_d(\boldsymbol{q}) + n(g) \quad \text{with} \quad \langle n(g) \rangle = 0, \quad \left\langle n^2(g) \right\rangle = \sigma_n^2(g) \tag{8.4}$$

A detailed treatment of noise in various types of imaging sensors can be found in Section 5.5; see also CVA1 [Chapter 9 and 10].

8.2.3 Multichannel signals

So far, only scalar signals have been considered. If more than one signal is taken simultaneously, a *multichannel signal* is obtained. In some cases, for example, taking time series at different spatial positions, the multichannel signal can be considered as just a sampled version of a higher-dimensional signal. In other cases, the individual signals cannot be regarded as samples. This is the case when they are parameters with different units and/or meaning.

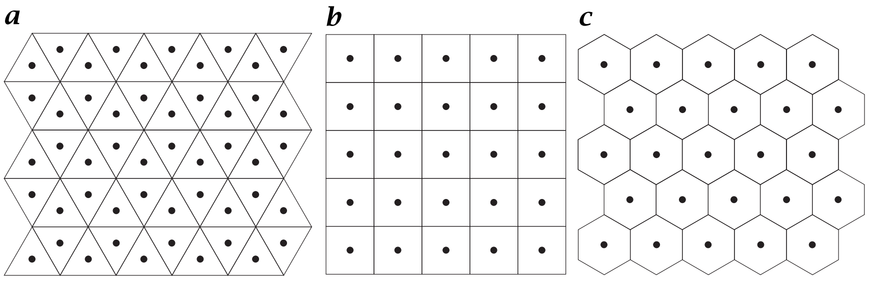

Figure 8.1: *Representation of 2-D digital images by meshes of regular polygons:* ***a*** *triangles;* ***b*** *squares;* ***c*** *hexagons.*

Table 8.2: *Properties of tessellations of the 2-D space with regular triangular, square, and hexagonal meshes; N_e: number of neighbors with common edge; N_c: number of neighbors with common edge and/or corner; l: basis length l of regular polygon; d: distance d to nearest neighbor; and A: area of cell*

	Triangular	Square	Hexagonal
N_e	3	4	6
N_c	12	8	6
l	$l = \sqrt{3}d = \sqrt{\sqrt{16/3}A}$	$l = d = \sqrt{A}$	$l = \frac{1}{3}\sqrt{3}d = \sqrt{\sqrt{4/27}A}$
d	$d = \frac{1}{3}\sqrt{3}l = \sqrt{\sqrt{16/27}A}$	$d = l = \sqrt{A}$	$d = \sqrt{3}l = \sqrt{\sqrt{4/3}A}$
A	$A = \frac{3}{4}\sqrt{3}d^2 = \frac{1}{4}\sqrt{3}l^2$	$A = d^2 = l^2$	$A = \frac{1}{2}\sqrt{3}d^2 = \frac{3}{2}\sqrt{3}l^2$

A multichannel signal provides a vector at each point and is therefore sometimes denoted as a *vectorial signal* and written as

$$\boldsymbol{g}(\boldsymbol{q}) = [q_1(\boldsymbol{q}), q_2(\boldsymbol{q}), \ldots, q_D(\boldsymbol{q})]^T \qquad (8.5)$$

A multichannel signal is not necessarily a vectorial signal. Depending on the mathematical relation between its components, it could also be a higher-order signal, for example, a *tensorial signal*. Such types of multichannel images are encountered when complex features are extracted from images. One example is the tensorial description of local structure discussed in Section 9.8.

8.3 Discrete signals

8.3.1 Regular two-dimensional lattices

Computers cannot handle continuous signals but only arrays of digital numbers. Thus it is required to represent signals as D-dimensional arrays of points. We first consider images as 2-D arrays of points. A

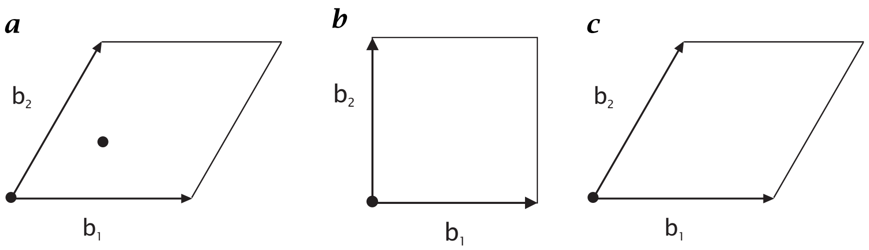

Figure 8.2: *Elementary cells of regular grids for 2-D digital images:* ***a*** *triangle grid;* ***b*** *square grid;* ***c*** *hexagonal grid.*

point on the 2-D grid is called a *pixel* or *pel*. Both words are abbreviations of *picture element*. A pixel represents the irradiance at the corresponding grid position. There are two ways to derive 2-D lattices from continuous signals.

First, the continuous 2-D space can be partitioned into space-filling cells. For symmetry reasons, only *regular polygons* are considered. Then there are only three possible *tesselations* with regular polygons: triangles, squares, and hexagons as illustrated in Fig. 8.1 (see also Table 8.2). All other regular polygons do not lead to a space-filling geometrical arrangement. There are either overlaps or gaps. From the mesh of regular polygons a 2-D array of points is then formed by the symmetry centers of the polygons. In case of the square mesh, these points lay again on a square grid. For the hexagonal mesh, the symmetry centers of the hexagons form a triangular grid. In contrast, the symmetry centers of the triangular grid form a more complex pattern, where two triangular meshes are interleaved. The second mesh is offset by a third of the base length l of the triangular mesh.

A second approach to regular lattices starts with a *primitive cell*. A primitive cell in 2-D is spanned by two not necessarily orthogonal base vectors $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$. Thus, the primitive cell is always a parallelogram except for square and rectangular lattices (Fig. 8.2). Only in the latter case are the base vectors $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ orthogonal. Translating the primitive cell by multiples of the base vectors of the primitive cell then forms the lattice. Such a *translation vector* or *lattice vector* $\boldsymbol{r}$ is therefore given by

$$\boldsymbol{r} = n_1\boldsymbol{b}_1 + n_2\boldsymbol{b}_2 \quad n_1, n_2 \in \mathbb{Z} \tag{8.6}$$

The primitive cells of the square and hexagonal lattices (Fig. 8.2b and c) contains only one grid located at the origin of the primitive cell. This is not possible for a triangular grid, as the lattice points are not arranged in *regular* distances along two directions (Fig. 8.1a). Thus, the construction of the triangular lattice requires a primitive cell with

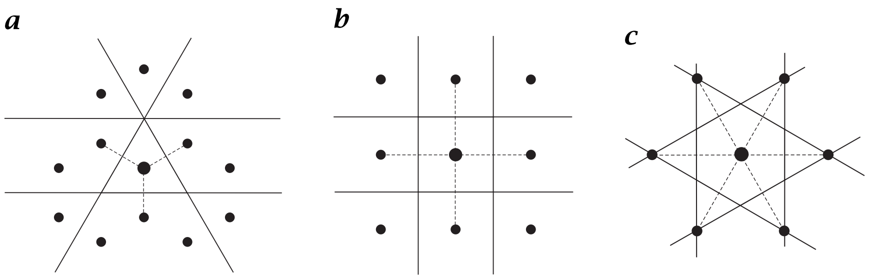

Figure 8.3: *Construction of the cells of a regular lattice from the lattice points:* ***a*** *triangle lattice;* ***b*** *square lattice; and* ***c*** *hexagonal lattice.*

two grid points. One grid point is located at the origin of the cell, the other is offset by a third of the length of each base vector (Fig. 8.2a)

The construction scheme to generate the elementary cells of regular shape from the lattice points is illustrated in Fig. 8.3. From one lattice point straight lines are drawn to all other lattice points starting with the nearest neighbors (dashed lines). Then the smallest cell formed by the lines perpendicular to these lines and dividing them into two halves results in the primitive cell. For all three lattices, only the nearest neighbors must be considered for this construction scheme.

The mathematics behind the formation of regular lattices in two dimensions is the 2-D analog to 3-D lattices used to describe crystals in solid state physics and mineralogy. The primitive cell constructed from the lattice points is, for example, known in solid state physics as the *Wigner-Seitz cell*.

Although there is a choice of three lattices with regular polygons—and many more if irregular polygons are considered—almost exclusively square or rectangular lattices are used for 2-D digital images.

The position of the pixel is given in the common notation for matrices. The first index m denotes the position of the row, the second, n, the position of the column (Fig. 8.4a); M gives the number of rows, and N the number of columns. In accordance with the matrix notation, the vertical axis (y axis) runs from top to bottom and not vice versa as is common in graphs. The horizontal axis (x axis) runs as usual from left to right.

8.3.2 Regular higher-dimensional lattices

The considerations in the previous section can be extended to higher dimensions. In 3-D space, lattices are identical to those used in solidstate physics to describe crystalline solids. In higher dimensions, we have serious difficulty in grasping the structure of discrete lattices because we can visualize only projections onto 2-D space. Given the fact that

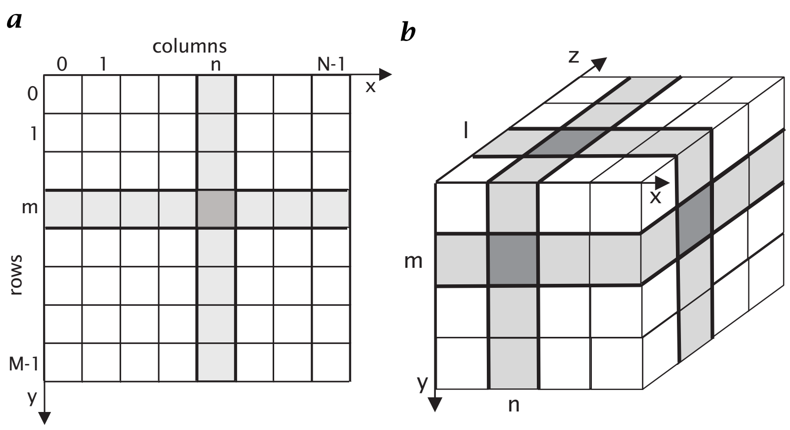

Figure 8.4: *Representation of digital images by orthogonal lattices:* ***a*** *square lattice for a 2-D image; and* ***b*** *cubic lattice for a volumetric or 3-D image.*

already 2-D discrete images are almost exclusively represented by rectangular lattices (Section 8.3.1), we may ask what we lose if we consider only hypercubic lattices in higher dimensions. Surprisingly, it turns out that this lattice has significant advantages. Thus it is hardly necessary to consider any other lattice.

Orthogonal lattice. The base vectors of the hypercubic primitive cell are orthogonal to each other. As discussed in CVA1 [Chapter 6], this is a significant advantage for the design of filters. If separable filters are used, they can easily be extended to arbitrary dimensions.

Valid for all dimensions. The hypercubic lattice is the most general solution for digital data as it is the only geometry that exists in arbitrary dimensions. In practice this means that it is generally quite easy to extend image processing algorithms to higher dimensions. We will see this, for example, with the discrete Fourier transform in Section 8.7, with multigrid data structures in Section 8.10, with averaging in Section 9.5, and with the analysis of local structure in Section 9.8.

Only lattice with regular polyhedron. While in 2-D three lattices with regular polyhedrons exist (Section 8.3.1), the cubic lattice is the only lattice with a regular polyhedron (the hexahedron) in 3-D. None of the other four regular polyhedra (tetrahedron, octahedron, dodecahedron, and icosahedron) is space filling.

These significant advantages of the hypercubic lattice are not outweighed by the single disadvantage that the neighborhood relations, discussed in Section 8.3.4, are more complex on these lattices than, for example, the 2-D hexagonal lattice.

In 3-D or *volumetric images* the elementary cell is known as a *voxel*, an abbreviation of *volume element*. On a rectangular grid, each voxel represents the mean gray value of a cuboid. The position of a voxel is given by three indices. The first, l, denotes the depth, m the row, and n the column (Fig. 8.4b). In higher dimensions, the elementary cell is denoted as a *hyperpixel*.

8.3.3 Metric in digital images

Based on the discussion in the previous two sections, we will focus in the following on hypercubic or orthogonal lattices and discuss in this section the metric of discrete images. This constitutes the base for all length, size, volume, and distance measurements in digital images. It is useful to generalize the *lattice vector* introduced in Eq. (8.6) that represents all points of a D-dimensional digital image and can be written as

$$\boldsymbol{r_n} = [n_1\Delta x_1, n_2\Delta x_2, \ldots, n_D\Delta x_D]^T \tag{8.7}$$

In the preceding equation, the lattice constants Δx_d need not be equal in all directions. For the special cases of 2-D images, 3-D volumetric images, and 4-D spatiotemporal images the lattice vectors are

$$\boldsymbol{r}_{m,n} = \begin{bmatrix} n\Delta x \\ m\Delta y \end{bmatrix}, \boldsymbol{r}_{l,m,n} = \begin{bmatrix} n\Delta x \\ m\Delta y \\ l\Delta z \end{bmatrix}, \boldsymbol{r}_{k,l,m,n} = \begin{bmatrix} n\Delta x \\ m\Delta y \\ l\Delta z \\ k\Delta t \end{bmatrix} \tag{8.8}$$

To measure distances, the *Euclidean distance* can be computed on an orthogonal lattice by

$$d_e(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{x} - \boldsymbol{x}'\| = \left[\sum_{d=1}^{D} (n_d - n'_d)^2 \Delta x_d^2\right]^{1/2} \tag{8.9}$$

On a square lattice, that is, a lattice with the same grid constant in all directions, the Euclidean distance can be computed more efficiently by

$$d_e(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{x} - \boldsymbol{x}'\| = \left[\sum_{d=1}^{D} (n_d - n'_d)^2\right]^{1/2} \Delta x \tag{8.10}$$

The Euclidean distance on discrete lattices is somewhat awkward. Although it is a discrete quantity, its values are not integers. Moreover, it cannot be computed very efficiently.

Therefore, two other metrics are sometimes considered in image processing. The *city-block distance*

$$d_b(\boldsymbol{x}, \boldsymbol{x}') = \sum_{d=1}^{D} |n_d - n'_d| \tag{8.11}$$

simply adds up the magnitude of the component differences of two lattice vectors and not the squares as with the Euclidean distance in Eq. (8.10). Geometrically, the city block distance gives the length of a path between the two lattice vectors if we can only walk in directions parallel to axes. The *chessboard distance* is defined as the maximum of the absolute difference between two components of the corresponding lattice vectors:

$$d_c(\boldsymbol{x}, \boldsymbol{x}') = \max_{d=1,\ldots,D} |n_d - n'_d| \tag{8.12}$$

These two metrics have gained some importance for morphological operations (Section 14.2.4). Despite their simplicity they are not of much use as soon as lengths and distances are to be measured. The Euclidean distance is the only metric on digital images that preserves the *isotropy* of the continuous space. With the city block and chessboard distance, distances in the direction of the diagonals are longer and shorter than the Euclidean distance, respectively.

8.3.4 Neighborhood relations

The term *neighborhood* has no meaning for a continuous signal. How far two points are from each other is simply measured by an adequate metric such as the Euclidean distance function and this distance can take any value. With the cells of a discrete signal, however, a ranking of the distance between cells is possible. The set of cells with the smallest distance to a given cell are called the nearest neighbors. The triangular, square, and hexagonal lattices have three, four, and six nearest neighbors, respectively (Fig. 8.5). The figure indicates also the ranking in distance from the central cell.

Directly related to the question of neighbors is the term *adjacency*. A *digital object* is defined as a *connected region*. This means that we can reach any cell in the region from any other by walking from one neighboring cell to the next. Such a walk is called a *path*.

On a square lattice there are two possible ways to define neighboring cells (Fig. 8.5b). We can regard pixels as neighbors either when they have a joint edge or when they have at least one joint corner. Thus a pixel has four or eight neighbors and we speak of a *4-neighborhood* or an *8-neighborhood*. The definition of the 8-neighborhood is somewhat awkward, as there are neighboring cells with different distances.

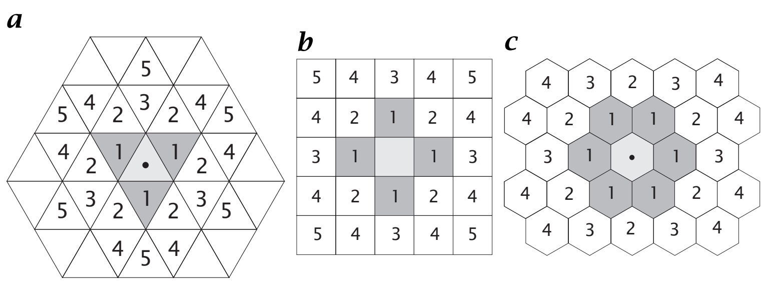

*Figure 8.5: Classification of the cells according to the distance from a given cell for the **a** triangular, **b** square, and **c** hexagonal lattices. The central cell is shaded in light gray, the nearest neighbors in darker gray. The numbers give the ranking in distance from the central cell.*

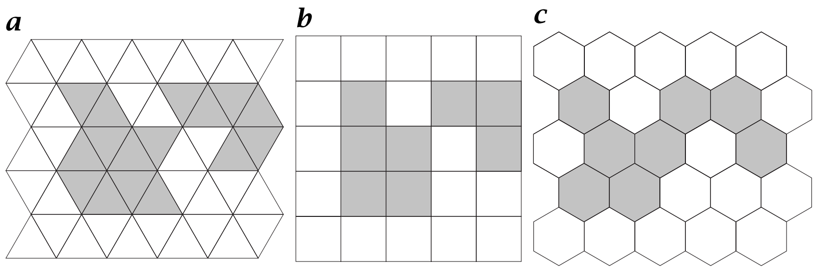

*Figure 8.6: Digital objects on **a** triangular, **b** square, and **c** hexagonal lattice; **a** and **b** show either two objects or one object (connected regions) depending on the neighborhood definition.*

The triangular lattice shows an equivalent ambivalence with the 3- and 12-neighborhoods with cells that have either only a joint edge or at least a joint corner with the central cell (Fig. 8.5a). In the 12-neighborhood there are three different types of neighboring cells, each with a different distance (Fig. 8.5a).

Only the hexagonal lattice gives a unique definition of neighbors. Each cell has six neighboring cells at the same distance joining one edge and two corners with the central cell.

A closer look shows that unfortunately both types of neighborhood definitions are required on triangular and square grids for a proper definition of connected regions. A region or an object is called connected when we can reach any pixel in the region by walking from one neighboring pixel to the next. The black object shown in Fig. 8.6b is one object in the 8-neighborhood, but constitutes two objects in the 4-neighborhood. The white background, however, shows the same property. Thus we have either two connected regions in the 8-neighborhood crossing each other or four separated regions in the 4-neighborhood.

This inconsistency between objects and background can be overcome if we declare the objects as 4-neighboring and the background as 8-neighboring, or vice versa.

These complications occur also on a triangular lattice (Fig. 8.6b) but not on a hexagonal lattice (Fig. 8.6c). The photosensors on the retina in the human eye, however, have a more hexagonal shape, see Wandell [1, Fig. 3.4, p. 49].

8.3.5 Errors in object position and geometry

The tessellation of space in discrete images limits the accuracy of the estimation of the position of an object and thus all other geometrical quantities such as distance, area, circumference, and orientation of lines. It is obvious that the accuracy of the position of a single point is only in the order of the lattice constant. The interesting question is, however, how this error propagates into position errors for larger objects and other relations. This question is of significant importance because of the relatively low spatial resolution of images as compared to other measuring instruments. Without much effort many physical quantities such as frequency, voltage, and distance can be measured with an accuracy better than 1 ppm, that is, 1 in 1,000,000, while images have a spatial resolution in the order of 1 in 1000 due to the limited number of pixels. Thus only highly accurate position estimates in the order of 1/100 of the pixel size result in an accuracy of about 1 in 100,000.

The discussion of position errors in this section will be limited to orthogonal lattices. These lattices have the significant advantage that the errors in the different directions can be discussed independently. Thus the following discussion is not only valid for 2-D images but any type of multidimensional signals and we must consider only one component.

In order to estimate the accuracy of the position estimate of a single point it is assumed that all positions are equally probable. This means a constant probability density function in the interval Δx. Then the variance σ_x^2 introduced by the position discretization is given by Papoulis [2, p. 106]

$$\sigma_x^2 = \frac{1}{\Delta x} \int\limits_{x_n-\Delta x/2}^{x_n+\Delta x/2} (x - x_n)^2 \, \mathrm{d}x = \frac{(\Delta x)^2}{12} \qquad (8.13)$$

Thus the standard deviation σ_x is about $1/\sqrt{12} \approx 0.3$ times the lattice constant Δx. The maximum error is, of course, $0.5\Delta x$.

All other errors for geometrical measurements of segmented objects can be related to this basic position error by statistical error propagation. We will illustrate this with a simple example computing the area and center of gravity of an object. For the sake of simplicity, we start with the unrealistic assumption that any cell that contains even the smallest fraction of the object is regarded as a cell of the object. We further assume that this segmentation is exact, that is, the signal itself does not contain noise and separates without errors from the background. In this way we separate all other errors from the errors introduced by the discrete lattice.

The area of the object is simply given as the product of the number N of cells and the area A_c of a cell. This simple estimate is, however, biased towards a larger area because the cells at the border of the object are only partly covered by the object. In the mean, half of the border cells are covered. Hence an unbiased estimate of the area is given by

$$A = A_c(N - 0.5N_b) \tag{8.14}$$

where N_b is the number of border cells. With this equation, the variance of the estimate can be determined. Only the statistical error in the area of the border cells must be considered. According to the laws of error propagation with independent random variables, the variance of the area estimate σ_A^2 is given by

$$\sigma_A^2 = 0.25A_c^2 N_b \sigma_x^2 \tag{8.15}$$

If we assume a compact object, for example, a square, with a length of D pixels, it has D^2 pixels and $4D$ border pixels. Using $\sigma_x \approx 0.3$ (Eq. (8.13)), the absolute and relative standard deviation of the area estimate are given by

$$\sigma_A \approx 0.3A_c\sqrt{D} \quad \text{and} \quad \frac{\sigma_A}{A} \approx \frac{0.3}{D^{3/2}} \quad \text{if } D \gg 1 \tag{8.16}$$

Thus the standard deviation of the area error for an object with a length of 10 pixels is just about the area of the pixel and the relative error is about 1 %. Equations (8.14) and (8.15) are also valid for volumetric images if the area of the elementary cell is replaced by the volume of the cell. Only the number of border cells is now different. If we again assume a compact object, for example, a cube, with a length of D, we now have D^3 cells in the object and $6D^2$ border cells. Then the absolute and relative standard deviations are approximately given by

$$\sigma_V \approx 0.45V_c D \quad \text{and} \quad \frac{\sigma_V}{V} \approx \frac{0.45}{D^2} \quad \text{if } D \gg 1 \tag{8.17}$$

Now the standard deviation of the volume for an object with a diameter of 10 pixels is about 5 times the volume of the cells but the relative error is about 0.5 %. Note that the absolute/relative error for volume measurements in/decreases faster with the size of the object than for area measurements.

The computations for the error of the center of gravity are quite similar. With the same assumptions about the segmentation process, an unbiased estimate of the center of gravity is given by

$$\boldsymbol{x}_g = \frac{1}{N}\left(\sum_{n=1}^{N-N_b}\boldsymbol{x}_n + \frac{1}{2}\sum_{n'=1}^{N_b}\boldsymbol{x}_{n'}\right) \tag{8.18}$$

Again the border pixels are counted only half. As the first part of the estimate with the nonborder pixels is exact, errors are caused only by the variation in the area of the border pixels. Therefore the variance of the estimate for each component of the center of gravity is given by

$$\sigma_g^2 = \frac{N_b}{4N^2}\sigma^2 \tag{8.19}$$

where σ is again the variance in the position of the fractional cells at the border of the object. Thus the standard deviation of the center of gravity for a compact object with the diameter of D pixels is

$$\sigma_g \approx \frac{0.3}{D^{3/2}} \quad \text{if } D \gg 1 \tag{8.20}$$

Thus the standard deviation for the center of gravity of an object with 10 pixel diameter is only about 0.01 pixel. For a volumetric object with a diameter of D pixel, the standard deviation becomes

$$\sigma_{gv} \approx \frac{0.45}{D^2} \quad \text{if } D \gg 1 \tag{8.21}$$

This result clearly shows that the position of objects and all related geometrical quantities such as the distances can be performed even with binary images (segmented objects) well into the range of 1/100 pixel. It is interesting that the relative errors for the area and volume estimates of Eqs. (8.16) and (8.17) are equal to the standard deviation of the center of gravity Equations (8.20) and (8.21). Note that only the *statistical error* has been discussed. A bias in the segmentation might easily result in much higher systematic errors.

8.4 Relation between continuous and discrete signals

A continuous function $g(\boldsymbol{q})$ is a useful mathematical description of a signal as discussed in Section 8.2. Real-world signals, however, can only be represented and processed as discrete or digital signals. Therefore a detailed knowledge of the relation between these two types of signals is required. It is not only necessary to understand the whole chain of the image-formation process from a continuous spatial radiance distribution to a digital image but also to perform subpixel-accurate image